Syntactical Rules for Colored Petri Nets Manipulation

Sami Evangelista

CEDRIC - CNAM Paris
292, rue St Martin, 75003 Paris
evangeli@cnam.fr

Abstract. Defining structural analysis techniques for colored Petri nets or generalizing existing techniques of ordinary nets to colored nets is made difficult by the management of the color mappings of the net. Indeed, the structure of the colored net does not necessarily reflect the one of the underlying net. A solution is to unfold the net and work directly on the unfolded net. Another one is to work directly on the colored net in a symbolic way. We explore in this work a symbolic framework based on constraints systems. A class of colored Petri nets is defined. We define simple rules to translate each color mapping allowed by this class into an equivalent constraints system. At last we work at a syntactical level to check properties of mappings. Two relevant examples of application are given.

1 Introduction

Colored nets provide designers a high level language to model concurrent systems. The price to pay is an increased difficulty of the possible analysis techniques. A typical example is the computation of flows of colored nets. As the structure of the colored net is not necessarily the one of its unfolded net, generalizing existing techniques of ordinary nets such as net reductions and stubborn sets to colored nets can be quite tricky.
There are two possible ways to deal with this additional difficulty. A first one is to unfold the net and work directly on it. This has two major drawbacks. Firstly, it needs to fix the parameters of the system. Secondly, color domains can be far too large to enable a complete enumeration. In our sense working with colored nets does not make sense if at some point the net has to be unfolded. All the interest of the colored nets theory is to define techniques applicable directly on colored nets without unfolding.
Another solution is two work at the colored net level in a symbolic way. In a previous work ([10]), we gave simple algorithms to compute a large set of color mappings and to check property of color mappings of a sub class of well formed nets. Indeed, the "good" structuring of mappings of well formed nets enable such algorithms. Our aim is now to define a more general symbolic technique which cover a larger set of colored nets classes.
Our work is inspired from a previous work of Brgan and Poitrenaud [9]. In this

paper, they propose to compute the stubborn sets of well formed nets in a symbolic manner by using constraints systems. Starting from this initial idea, we build a general framework, which enable to compute a large set of color mappings symbolically.
We focus on a class of safe colored nets which allows any user defined mapping. For each mapping construction allowed by this class we show how it can be translated into an equivalent constraints system. The usefulness of this work is illustrated by two examples : pre agglomeration ([4], [5], [6], [12], [13]) and conflicts resolution ([2], [3]).
This work is organized as follow. Section 2 gives some basic definitions on Petri nets and colored Petri nets. Section 3 presents our framework. At last section 4 gives two examples of application for our method.

2 Basic definitions

We recall here some basic definitions on ordinary Petri nets and colored nets. We also present the sub class of colored nets studied in this work.

Note. The set of booleans is noted $\mathbb{B}$. The set of natural numbers is noted $\mathbb{N}$. If E is a set, then $\mathcal{P}(E)$ is the set of sub sets of E, i.e, $e \in \mathcal{P}(E) \Leftrightarrow e \subseteq E$.

2.1 Petri nets

The nets we study are safe, i.e, each place can contain at most one token. Therefore we define markings of places as boolean values indicating whether or not a token is in the place. Similarly incidence matrixes indicate whether or not a token is consumed (or put).

Definition 1. *A safe net is a tuple* $\langle P, T, W^-, W^+, m_0 \rangle$ *where :*

- P *is a finite set of places*
- T *is a finite set of transitions such that* $P \cap T = \emptyset$
- W^- *and* W^+*, the backward and forward incidence matrixes are mappings from* $P \times T$ *to* $\mathbb{B}$
- m_0 *is a mapping from* P *to* $\mathbb{B}$

Petri nets are graphically represented by drawing places as circles, transitions as boxes (or bars). Incidence matrixes define arcs (and the mapping labeling the arc for colored nets). Initial marking indicates tokens in places. These tokens are represented by dots for ordinary nets and tuples for colored Petri nets.
The set of input (resp. output) places of a transition t is the set $\{p \in P \mid W^-(p,t)\}$ (resp. $\{p \in P \mid W^+(p,t)\}$). Similarly, the set of input (resp. output) transitions of a place p is the set $\{t \in T \mid W^+(p,t)\}$ (resp. $\{t \in T \mid W^-(p,t)\}$). These definitions are extended to set of places and transitions, i.e., for $P' \in \mathcal{P}(P)$, ${}^\bullet P' = \cup_{p \in P'} {}^\bullet p$

2.2 Safe Colored Petri nets

In colored Petri nets, tokens contained in places have colors, and places and transitions have color domains. Markings of safe colored nets are subsets of places domains.

Definition 2. *A safe colored Petri net is a tuple* $\langle P, T, C, W^-, W^+, m_0 \rangle$ *where :*

- P *is a finite set of places*
- T *is a finite set of transitions such that* $P \cap T = \emptyset$
- $C : (P \cup T) \rightarrow \omega$ *is a color domain mapping.* ω *is the finite set of finite and non empty sets.*
- W^- *is a mapping from* $P \times T$ *such that* $\forall p \in P, t \in T, W^-(p,t)$ *is a mapping from* $C(t)$ *to* $\mathcal{P}(C(p))$
- W^+ *is a mapping from* $P \times T$ *such that* $\forall p \in P, t \in T, W^+(p,t)$ *is a mapping from* $C(t)$ *to* $\mathcal{P}(C(p))$
- m_0 *is a mapping from* P *such that* $\forall p \in P, m_0(p)$ *is an element of* $\mathcal{P}(C(p))$

As said previously, a colored net is equivalent to an ordinary net obtained by an unfolding operation. The hypothesis of finiteness of color domains is necessary for the unfolding.

Definition 3 (Unfolding). *Let* $N = \langle P, T, C, W^-, W^+, m_0 \rangle$ *be a colored net.* $\langle P_u, T_u, {W^-}_u, {W^+}_u, {m_0}_u \rangle$*, the ordinary net obtained by the unfolding of* N *is such that :*

- $P_d = \cup_{p \in P, c \in C(p)}(p, c)$
- $T_d = \cup_{t \in T, c \in C(t)}(t, c)$
- $\forall p \in P, t \in T, c_p \in C(p), c_t \in C(t),$
  ${W^-}_u((p, c_p), (t, c_t)) \Leftrightarrow c_p \in W^-(p,t)(c_t)$
  ${W^+}_u((p, c_p), (t, c_t)) \Leftrightarrow c_p \in W^+(p,t)(c_t)$
- $\forall p \in P, c \in C(p), {m_0}_u((p, c)) \Leftrightarrow c \in m_0(p)$

In the remainder a place (p, c) obtained by the unfolding of place p is called an instance of p. Similarly, a transition (t, c) obtained by the unfolding of transition t is called an instance of t. The term *underlying net* is also used to denote the unfolded net.

2.3 A sub class of safe Colored Petri nets

Color domains of places and transitions of the studied class of colored nets are cartesian products of basic set called color classes.

Definition 4 (Color domain). *Let* Cl *be a finite set of finite sets* $\{C_1, \ldots, C_{|Cl|}\}$*. A color domain* C *is a mapping from* $[1..ar(C)]$ *to* Cl *where* ar *is a function from* $\mathbb{C}$*, the set of color domains to* $\mathbb{N}$*. A color domain* C *is noted* $\langle C(1), \ldots, C(ar(e)) \rangle$

Color mappings are constructed over a set of elementary mappings. This elementary mappings are the projection, a constant, the broadcast or a user defined mapping. This elementary mappings can be combined to form tuples. At last color mappings are combinations of tuples.

A projection mapping select a specific element in a tuple.

Definition 5 (Projection mapping). *Let C be a color domain and C_i be a color class. The set of projections from C to $\mathcal{P}(C_i)$ is $\{X_i \mid C(i) = C_i\}$. For each X_i, $c = \langle c_1, \ldots, c_{ar(C)} \rangle \in C$, $X_i(c) = \{c_i\}$. The set of projections is noted $Proj_{C \to C_i}$.*

A constant mapping always select the same element.

Definition 6 (Constant mapping). *Let C be a color domain and C_i be a color class. The set of constant mappings from C to $\mathcal{P}(C_i)$ is $\{\lambda \mid \lambda \in C_i\}$. For each λ, $c \in C$, $\lambda(c) = \{\lambda\}$. The set of constant mappings is noted $Const_{C \to C_i}$.*

The broadcast mapping select all the element of the target color class.

Definition 7 (Broadcast mapping). *Let C be a color domain and C_i be a color class. The set of broadcast mappings from C to $\mathcal{P}(C_i)$ is the singleton $\{All\}$. For each $c \in C$, $All(c) = \{c' \in C_i\}$. The set of broadcast mappings is noted $Broadcast_{C \to C_i}$*

At last we allow any user-defined function. Each function f is defined from $C_1^f \times \cdots \times C_{n_f}^f$ to a color class C_i. Parameters of a function can be any elementary mapping except the broadcast mapping (as this mapping produces several tokens, whereas others elementary mappings can be interpreted as mapping from C to C_i since they produce a single token).

Definition 8 (User defined mapping). *Let C be a color domain and C_i be a color class. Let $\mathbb{F}_i$ be the set of mappings defined to C_i. The domain of each $f \in \mathbb{F}_i$ is $C_1^f \times \cdots \times C_{n_f}^f$ The set of user mappings from C to $\mathcal{P}(C_i)$ is the set of couples*
$\{(f, (c_1^f, \ldots, c_{n_f}^f)) \mid f \in \mathbb{F}_i \ \wedge \ \forall c_i^f, c_i^f \in Const_{C \to C_i^f} \cup Proj_{C \to C_i^f} \cup User_{C \to C_i^f}\}$.
For each $c \in C$, $(f, (c_1^f, \ldots, c_{n_f}^f))(c) = \{f(c_1^f, \ldots, c_{n_f}^f)\}$. The set of user mappings from C to $\mathcal{P}(C_i)$ is noted $User_{C \to C_i}$.

Projections, constants, broadcast, and user defined mappings form the set of elementary mappings.

Definition 9 (Elementary mapping). *Let C be a color domain and C_i be a color class. The set of elementary function from C to $\mathcal{P}(C_i)$ is the set*
$Elem_{C \to C_i} = Proj_{C \to C_i} \cup Const_{C \to C_i} \cup Broadcast_{C \to C_i} \cup User_{C \to C_i}$

A tuple is a set of elementary functions.

Definition 10 (Tuple). *Let C and C' be two color domains. A tuple tup is a set $\cup_{i \in [1..ar(C')]} f_i \in Elem_{C \to C'}$. tup is a mapping from C to $\mathcal{P}(C')$ defined by :*
$\forall c \in C, c' = \langle c'_1, \ldots, c'_{ar(C')} \rangle \in C'$,

$c' \in t(c) \Leftrightarrow c'_1 \in f_1(c) \wedge \cdots \wedge c'_{ar(C')} \in f_{ar(C')}(c)$.
We note $Tup_{C \to C'}$ *the set of tuples from* C *to* $\mathcal{P}(C')$. *A tuple is also noted* $\langle f_1, \ldots, f_{ar(C')} \rangle$.

At last a color mapping is defined recursively as a tuple, a union of mappings, or an intersection of mappings.

Definition 11 (Color mapping). *Let* C *and* C' *be two color domains.* $Map_{C \to C'}$, *the set of color mapping from* C *to* $\mathcal{P}(C')$ *is :*

- $t \in Tup_{C \to C'}$
- $m \cup m'$ *with* $m, m' \in Map_{C \to C'}$
- $m \cap m'$ *with* $m, m' \in Map_{C \to C'}$

3 A symbolic framework

Our framework is inspired from an initial idea of Poitrenaud and Brgan on stubborn sets. In [9] they propose to compute stubborn sets of well formed nets by means of constraints systems solving. For instance if we consider the mapping $f(\langle X, Y, Z \rangle) = \langle Y, 2, All \rangle$, the set of colors $\langle X', Y', Z' \rangle$ which belongs to the application $f(\langle X, Y, Z \rangle)$ satisfy the constraints system $X' = Y \wedge Y' = 2$. For each mapping construction (union, difference, ...) allowed by the class of nets under study we give an equivalent constraints system. To be able to define transposition and composition we extend color mappings domain to powerset in a natural way : if f is a mapping from C to $\mathcal{P}(C')$, $f_{\mathcal{P}(C)\to}$ is a mapping from $\mathcal{P}(C)$ to $\mathcal{P}(C')$ defined by $f_{\mathcal{P}(C)\to}(C'') = \cup_{c \in C''} f(c)$. In the remainder, no distinction is done between f and $f_{\mathcal{P}(C)\to}$. Similarly, $Tup_{C \to C'}$ and $Map_{C \to C'}$ also denote respectively the set of tuples from $\mathcal{P}(C)$ to $\mathcal{P}(C')$ and the set of mappings from $\mathcal{P}(C)$ to $\mathcal{P}(C')$.
The definition of constraints systems is given below. Informally, such a system is a boolean combination of elementary predicates (also called basic predicates) on the variables of the system. Each variable is associated a color class of Cl by a mapping D. In addition, we distinguish two different set of variables : the in and out variables of the systems. These variables correspond to the domain and the codomain of the mapping which is represented.

Definition 12. *A constraints system is a tuple* $\langle V, D, I, O, P \rangle$ *where* V *is a set of variables,* D *is a mapping from* V *to* Cl, $I \subseteq V$ *is a set of in variables,* $O \subseteq V$ *is a set of out variables such that* $I \cap O = \emptyset$, *and* P *is a boolean combination of basic predicates over the set* V. $N = V \setminus (I \cup O)$ *is the set of intermediate variables.*

A basic predicate is any boolean condition under the form e_1 *op* e_2 where e_1 and e_2 are either a variable of the system either a constant, either a user defined mapping and *op* is in the set $\{=, \neq\}$.
Each constraints system is equivalent to a color mapping. A color $\langle c_1, \ldots, c_n \rangle$ is produced by the application $m(\langle c'_1, \ldots, c'_m \rangle)$ if we can find an assignment to the

intermediate variables such that the value of predicate P is evaluated to *true* for $I_1 = c_1, \ldots, I_n = c_n, O_1 = c'_1, \ldots, O_m = c'_m$.

Definition 13. *Let $C = \langle V, D, I, O, P\rangle$ be a constraints system such that $I = \{I_1, \ldots, I_n\}$, $O = \{O_1, \ldots, O_m\}$, $N = \{N_1, \ldots, N_l\}$. The equivalent color mapping ψ of C noted $\psi \equiv C$ is the mapping from $\mathcal{P}(\langle D(I_1), \ldots, D(I_n)\rangle)$ to $\mathcal{P}(\langle D(O_1), \ldots, D(O_m)\rangle)$ defined by :*

$$\langle o_1, \ldots, o_m\rangle \in \psi(\langle i_1, \ldots, i_n\rangle)$$
$$\Leftrightarrow$$
$$\exists n_1, \ldots, n_l \mid P(i_1, \ldots, i_n, n_1, \ldots, n_l, o_1, \ldots, o_m)$$

3.1 Building constraints systems

Elementary mappings Each elementary mapping is translated into an elementary predicate. If f is an elementary mapping, we note $Pred_f$ the corresponding elementary mapping. In order to define the predicate associated to an elementary mapping we have to know the tuple in which the elementary mapping appears to bind the correct variable.

Proposition 1. *Let C and C' be two color domains, and $t = \langle t_1, \ldots, t_{ar(C')}\rangle \in Tup_{C\to C'}$ such that $t \equiv \langle V, D, I, O, P\rangle$, $I = \{I_1, \ldots, I_{ar(C)}\}$ and $O = \{O_1, \ldots, O_{ar(C')}\}$. For each t_i the elementary predicate $pred_{t_i}$ associated to t_i is*

$$pred_{t_i} = \begin{cases} \text{if } t_i = X_j & \text{then } O_i = I_j \\ \text{if } t_i = \lambda & \text{then } O_i = \lambda \\ \text{if } t_i = All & \text{then } true \\ \text{if } t_i = (f, \langle c_1, \ldots, c_{n_f}\rangle) & \text{then } O_i = (f, \langle c'_1, \ldots, c'_{n_f}\rangle) \end{cases}$$

where each c'_i is obtained from c_i by replacing each X_k by I_k.

Tuples A sufficient and necessary condition for a color $\langle c_1, \ldots, c_n\rangle$ to be produced by a tuple $\langle f_1, \ldots, f_n\rangle$ applied to a color c is that each c_i belongs to the application $f_i(c)$. Therefore, the equivalent constraints system of a tuple is obtained by the conjunction of the basic predicate associated to the f_i.

Proposition 2. *Let C and C' be two color domains, and $t = \langle t_1, \ldots, t_m\rangle \in Tup_{C\to C'}$. If*

- $V = \{I_1, \ldots, I_{ar(C)}\} \cup \{O_1, \ldots, O_{ar(C')}\}$
- $\forall i \in [1..ar(C)], D(I_i) = C(i)$, *and* $\forall i \in [1..ar(C')], D(O_1) = C'(i)$
- $I = \{I_1, \ldots, I_{ar(C)}\}$
- $O = \{O_1, \ldots, O_{ar(C')}\}$
- $P = [pred_{t_1} \ \wedge \ \ldots \ \wedge \ pred_{t_{ar(C')}}]$

then $t \equiv \langle V, D, I, O, P\rangle$

Example 1. Let $t = \langle X, Y, max(X,Y), All, 0\rangle$ be a tuple from $\mathcal{P}(\langle C, C\rangle)$ to $\mathcal{P}(\langle C, C, C, C, C\rangle)$. The corresponding constraints system is
$V = \{I_1, I_2, O_1, O_2, O_3, O_4, O_5\}$
$\forall v \in V, D(v) = C$
$I = \{I_1, I_2\}$
$O = \{O_1, O_2, O_3, O_4, O_5\}$
$P = [O_1 = I_1 \wedge O_2 = I_2 \wedge O_3 = max(I_1, I_2) \wedge O_5 = 0]$
We note that I_4 does not appear in the predicate since the corresponding elementary mapping is the broadcast.

Union, intersection, and difference To deal with union, intersection and difference, we simply perform respectively the disjunction, the intersection, and the disjunction of the right negation of the predicates corresponding to the mappings.

Proposition 3. *Let C and C' be two color domains, $m_1 \in Map_{C\to C'}$ and $m_2 \in Map_{C\to C'}$. If*

- $m_1 \equiv \langle V_1, D_1, I, O, P_1\rangle$
- $m_2 \equiv \langle V_2, D_2, I, O, P_2\rangle$
- $V = V_1 \cup V_2$
- $\forall v_1 \in V_1, D(v_1) = D_1(v_1)$ *and* $\forall v_2 \in V_2, D(v_2) = D_1(v_2)$
- $P_\cup = [P_1 \vee P_2]$
- $P_\cap = [P_1 \wedge P_2]$
- $P_\setminus = [P_1 \wedge \neg P_2]$

then

- $m_1 \cup m_2 \equiv \langle V, D, I, O, P_\cup\rangle$
- $m_1 \cap m_2 \equiv \langle V, D, I, O, P_\cap\rangle$
- $m_1 \setminus m_2 \equiv \langle V, D, I, O, P_\setminus\rangle$

Example 2. Let $t = \langle X, Y, 0\rangle$ and $t' = \langle X, Y, f(X)\rangle$ be two tuples from $\mathcal{P}(\langle C, C\rangle)$ to $\mathcal{P}(\langle C, C, C\rangle)$. The constraints system obtained for $t \cup t'$, $t \cap t'$, $t \setminus t'$ are respectively $\langle V, D, I, O, P_\cup\rangle$, $\langle V, D, I, O, P_\cap\rangle$, and $\langle V, D, I, O, P_\setminus\rangle$ where :
$V = \{I_1, I_2, O_1, O_2, O_3\}$
$\forall v \in V, D(v) = C$
$I = \{I_1, I_2\}$
$O = \{O_1, O_2, O_3\}$
$P_\cup = [(O_1 = I_1 \wedge O_2 = I_2 \wedge O_3 = 0) \vee (O_1 = I_1 \wedge O_2 = I_2 \wedge O_3 = f(I_1))]$
$P_\cap = [O_1 = I_1 \wedge O_2 = I_2 \wedge O_3 = 0 \wedge O_1 = I_1 \wedge O_2 = I_2 \wedge O_3 = f(I_1)]$
$P_\cap = [(O_1 = I_1 \wedge O_2 = I_2 \wedge O_3 = 0) \wedge \neg(O_1 = I_1 \wedge O_2 = I_2 \wedge O_3 = f(I_1))]$

Transposition The transposition is often used to find instances of transitions which are linked to a place instance. For instance ${}^tW^-(p,t)(c_p)$ is the set of colors of c_t such that (t, c_t) consumes a token in (p, c_p). We give now its definition.

Definition 14 (Transposition). *If C and C' are color domain and f is a mapping from $\mathcal{P}(C)$ to $\mathcal{P}(C')$, then ${}^t f$ is a mapping from $\mathcal{P}(C')$ to $\mathcal{P}(C)$ defined by*
${}^t f(c') = \{c \in C \mid c' \in f(c)\}$

The constraints system of the transposition is simply obtained by switching the in and out set of variables.

Proposition 4. *Let C and C' be two color domains, $m \in Map_{C \to C'}$. If $m \equiv \langle V, D, I, O, P \rangle$ then ${}^t m \equiv \langle V, D, O, I, P \rangle$.*

Composition The composition operation allows to find instances of a place (resp. transition) which are linked to another place (transition) by an intermediate transition (place).

Definition 15 (Composition). *If C, C' and C'' are color domain, f is a mapping from $\mathcal{P}(C'')$ to $\mathcal{P}(C')$, and g is a mapping from $\mathcal{P}(C)$ to $\mathcal{P}(C'')$, then $f \circ g$ is a mapping from $\mathcal{P}(C)$ to $\mathcal{P}(C')$ defined by*
$(f \circ g)(c) = \{c' \in C' \mid \exists c'' \mid c'' \in g(c) \wedge c' \in f(c'')\}$

The constraints system associated to a composition $f \circ g$ is obtained by the conjunction of the systems of f and g. In addition, we have to link each output variable of g to the corresponding input variable of f.

Proposition 5. *Let C, C' and C'' be three color domains, $m_1 \in Map_{C'' \to C'}$ and $m_2 \in Map_{C \to C''}$. If*

- $m_1 \equiv \langle V_1, D_1, I_1, O_1, P_1 \rangle$ *with* $I_1 = \{I_{1,1}, \ldots, I_{1,ar(C'')}\}$
- $m_2 \equiv \langle V_2, D_2, I_2, O_2, P_2 \rangle$ *with* $O_2 = \{O_{2,1}, \ldots, O_{2,ar(C'')}\}$
- $V = V_1 \cup V_2$
- $\forall v_1 \in V_1, D(v_1) = D_1(v_1)$ *and* $\forall v_2 \in V_2, D(v_2) = D_1(v_2)$
- $I = I_2$
- $O = O_1$
- $P = P_1 \wedge P_2 \wedge_{j \in [1..ar(C'')]} I_{1,j} = O_{2,j}$

then

- $m_1 \circ m_2 \equiv \langle V, D, I, O, P \rangle$

Example 3. Let $m_1 = \langle X, 0 \rangle$ and $m_2 = \langle X, f(X) \rangle$ be two tuples respectively from $\mathcal{P}(\langle C, C \rangle)$ to $\mathcal{P}(\langle C, C \rangle)$ and from $\mathcal{P}(\langle C \rangle)$ to $\mathcal{P}(\langle C, C \rangle)$. The constraints system obtained for $m_1 \circ m_2$ is :
$V = \{I_{m_1,1}, I_{m_1,2}, O_{m_1,1}, O_{m_1,2}, I_{m_2,1}, O_{m_2,1}, O_{m_2,2}\}$
$\forall v \in V, D(v) = C$
$I = \{I_{m_2,1}\}$
$O = \{O_{m_1,1}, O_{m_1,2}\}$
$P = [\, O_{m_1,1} = I_{m_1,1} \wedge O_{m_1,2} = 0 \wedge$
$O_{m_2,1} = I_{m_2,1} \wedge O_{m_2,2} = f(I_{m_2,1}) \wedge$
$I_{m_1,1} = O_{m_2,1} \wedge I_{m_1,2} = O_{m_2,2}]$

Summary The following table summarizes the possible constraints systems.

Mapping	Predicates	In	Out
$\langle f_1, \ldots, f_m \rangle$	$pred_{f_1} \wedge \ldots \wedge pred_{f_m}$	$\{I_1, \ldots, I_n\}$	$\{O_1, \ldots, O_m\}$
$m_1 \cup m_2$	$P_{m_1} \vee P_{m_2}$	I_{m_1}	O_{m_1}
$m_1 \cap m_2$	$P_{m_1} \wedge P_{m_2}$	I_{m_1}	O_{m_1}
$m_1 \setminus m_2$	$P_{m_1} \wedge \neg P_{m_2}$	I_{m_1}	O_{m_1}
${}^t m_1$	P_{m_1}	O_{m_1}	I_{m_1}
$m_1 \circ m_2$	$P_{m_1} \wedge P_{m_2} \wedge_i I_{1i} = O_{2i}$	I_{m_2}	O_{m_1}

3.2 Exploiting constraints systems

Checking mapping properties is of course impossible in the general case, i.e. when the mapping contains some user defined mappings, as it is impossible to reverse such a mapping. However in some cases we are able to check some mapping properties at a syntactical level. In order to enable simple syntactical analysis we put the systems in a canonical form that we call reduced form. A system is in its reduced form if it is put in normal disjunctive form $\vee_i P_i$ and each P_i is reduced. We define some reduction rules that preserve the validity of the system.

Definition 16. *Let P be a conjunction of basic predicates. P_r, the reduced form of P is obtained by applying these rules until no more rule is applicable. We assume In and Out are respectively the input and output variables of the system.*

Transitivity rule *(suppress an intermediate variable used to link two others)*
if $P = [expr = X] \wedge [X = expr'] \wedge P'$ such that $X \notin In \cup Out$
then $P \leftarrow P' \wedge [expr = expr']$

Inconsistency rule *(two contradictory predicates appear in the conjunction)*
if $P = P_1 \wedge P_2 \wedge P'$ such that $P_1 = [X = a]$ and $P_1 = [X = b]$ or $P_1 = [expr = expr']$ and $P_2 = [expr \neq expr']$
then $P \leftarrow false$

Elimination rule *(suppress a useless intermediate variable)*
if $P = [expr = X] \wedge P'$ such that $X \notin In \cup Out \cup Var(P')$
then $P \leftarrow P'$

where X, Y, Z are variables, a, b are constants, and $expr, expr'$ are any expression allowed in a basic predicate. $Var(P')$ denotes the set of variables which appear in P'.

We give now an example of system reduction.

Example 4. Let $\langle V, D, I, O, P \rangle$ be a system such that :

- $V = \{i, j, k, l, m, n, o, p, q\}$
- $\forall v \in V, D(c) = \mathbb{N}$
- $I = \{i, j, k\}$
- $O = \{o, p, q\}$

– $P = [i = l \wedge l = o \wedge p = f(m) \wedge k = 2 \wedge q = n \vee$
$f(l) = o \wedge k = n \wedge n = 2 \wedge n = q \wedge q = 0]$

We can reduce the system by the following sequence of reductions applications :
transitivity rule on $i = l \wedge l = o \rightarrow i = o$
elimination rule on $q = n$
transitivity rule on $n = 2 \wedge n = q \rightarrow q = 2$
inconsistency rule on $q = 2 \wedge q = 0$

Finally, the fully reduced system is $P = [i = o \wedge p = f(m) \wedge k = 2]$

We give now two examples of such syntactical treatments : emptiness test and inclusion test.

Emptiness test Emptiness test consists in checking that a mapping m is such that $\forall c \in dom(c), m(c) = \emptyset$. To check emptiness it is sufficient to put the system of m in its reduced form and to check that it is reduced to $false$. An example is given in the next section.

Inclusion test Inclusion test consists in checking that two mappings m_1 and m_2 from C to $\mathcal{P}(C')$ are such that $\forall c \in C, m_2(c) \subseteq m_1(c)$. This relation is noted $\sqsubseteq$. It is sufficient to show that for each predicates conjunction c_2 of m_2 there is a predicates conjunction c_1 in m_1 such that each basic predicate of c_1 is also a basic predicate of c_2 by a mapping ψ which map each variable of c_1 to a variable of c_2.

Proposition 6. *Let m and m' be two mappings from $\mathcal{P}(C)$ to $\mathcal{P}(C')$. Let $\phi = \langle V, D, I, O, \vee_i P_i \rangle$ and $\phi' = \langle V', D', I, O, \vee_i P'_i \rangle$ be their respective reduced systems. $m \sqsubseteq m'$ if there is an injective application ψ from V' to V such that*

1. *$\forall v \in V', D'(v) = D(\psi(v))$, and $\forall v \in (I \cup O), \psi(v) = v$*
2. *$\forall P_i = \wedge_j P_{i,j}, \exists P'_{i'} = \wedge_{j'} P'_{i',j'}$, such that $\forall P'_{i',j'}, \exists P_{i,j} \mid \psi(P'_{i',j'}) \Leftrightarrow P_{i,j}$*

where $\psi(P'_{i',j'})$ maps basic predicate $P'_{i',j'}$ to the basic predicate obtained by replacing a variable v by $\psi(v)$

Below is an example of inclusion test.

Example 5. Let $m_1 \equiv [X = 0 \wedge A = Y \wedge B = g(E) \vee B = 0 \wedge A = g(F)]$ and $m_2 \equiv [X = 0 \wedge B = g(J) \vee A = g(H)]$ such that their input variables are X, Y and their output variables are A, B. We can state that $m_1 \sqsubseteq m_2$ since there is the mapping ψ defined by $\psi(J) = E, \psi(H) = F, \psi(A) = A, \psi(B) = B, \psi(X) = X, \psi(Y) = Y$ and such that :
$\psi([X = 0]) = [X = 0]$, $\psi([B = g(J)]) = [B = g(E)]$
and
$\psi([A = g(H)]) = [A = g(F)]$

4 Examples of application

We illustrate the usefulness of our work by two examples. In a first example we see how it can be used to generalize ordinary pre agglomerations to colored nets, then we show the usefulness of this work for checking conflicts between transitions in a symbolic way.

4.1 Ordinary pre agglomeration

Pre and post agglomerations ([4],[5],[6]) are structural reductions which enable to merge local transitions which do not involve synchronizations. They are very useful to tackle the state explosion problem. They preserve all basic properties of the net (liveness, boundness, ...), as well as many temporal properties ([4]) and their application conditions only relies on the structure of the net. They have been generalized by Haddad in [8] and then in [12] to colored nets.
The idea of the first one, is to merge a transition h with a set of transitions F. h can be viewed as a local transition which firing can be delayed. Sufficient structural conditions which ensure a correct transformation with respect to general properties are :

1. $\exists p \in P$ such that
   (a) ${}^\bullet p = \{h\}$
   (b) $h^\bullet = \{p\}$
   (c) $p^\bullet = \{F\}$
   (d) $m_0(p) = false$
2. $\forall q \in {}^\bullet h, q^\bullet = \{h\}$

The key point which ensures that firing of h can be delayed is point 2. Indeed, as h does not share its input places, its input places can not lost tokens, and thus its firing can be delayed. Let us give a colored version of this point. For a colored generalization to be valid, each place (p, c) in the underlying net must fulfill these conditions. In [12], conditions given to ensure that each (h, c_h) does not share its input place with another (t, c_t), are :
$\forall q \in {}^\bullet h$,

- $W^-(d, h)$ is a quasi-injective mapping, i.e., $\forall c_1 \in C, c_2 \in C, c' \in C', c' \in f(c_1) \wedge c' \in f(c_2) \Rightarrow c_1 = c_2$.
- $q^\bullet = \{h\}$

Instead of forbidding that a place q has no other transition than h, we could state the following : for each place (p, c_p) the set of instances (t, c_t) that can be reached via a transition (h, c_h) and a place (q, c_q) input of (h, c_h) and such that (t, c_t) is an output of (q, c_q) is empty. This set of instances is given by the following mapping :

$$\phi(c) = \cup_{q \in {}^\bullet h, t \in q^\bullet \setminus \{h\}} {}^tW^-(q,t) \circ W^-(q,h) \circ {}^tW^+(p,h)(c) = \emptyset$$

${}^tW^+(p,h)(c)$ is the set of instances of h which have (p, c_p) as a post condition. If we compose this set with $W^-(q,h)$ we get the instances (q, c_q) which are linked to (p, c_p) by a (h, c_h). At last by composing this mapping with ${}^tW^-(q,t)$, we get the instances (t, c') which share their input places with the instance (h, c_h) input of (p, c_p).
Such a condition can be checked by constructing the constraints system equivalent to the mapping and by checking if it contains inconsistencies. For instance, let us take net depicted on figure 1. This fragment of net could correspond to an if-then-else statement.

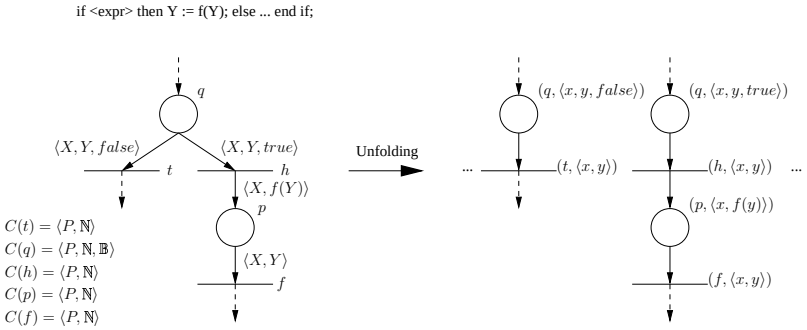

Fig. 1. Second condition is fulfilled for a colored pre agglomeration

As the only input place of h is q and $q^\bullet \setminus \{h\} = \{t\}$, the system $\langle V, D, I, O, P\rangle$ of ϕ is :

$$
\begin{array}{l}
V = \{X_p, Y_p, X_h, Y_h, X_q, Y_q, Z_q, X_t, Y_t\} \\
D(X_p) = D(X_h) = D(X_q) = D(X_t) = \mathbb{P}, \\
D(Y_p) = D(Y_h) = D(Y_q) = D(Y_t) = \mathbb{N}, \\
D(Z_q) = \mathbb{B} \\
I = \{X_p, Y_p\} \\
O = \{X_t, Y_t\} \\
P = \left\{ \begin{array}{lll}
X_p = X_h \wedge Y_p = f(Y_h) \wedge & & \\
X_q = X_h \wedge Y_q = Y_h & \wedge Z_q = true & \wedge \\
X_t = X_q \wedge Y_t = Y_q & \wedge Z_q = false &
\end{array} \right.
\end{array}
$$

We easily see that there is an inconsistency in the system ($Z_q = true \wedge Z_q = false$, so we can conclude that the second condition for a pre agglomeration of h with f is fulfilled. Indeed if we look at the underlying net, it appears that no instance (h, c_h) shares its input place, which is a necessary condition to ensure a correct agglomeration of h with f.

4.2 Resolution of conflicts

The definition of conflicts has been given by Dutheillet and Haddad in [2] and [3]. A conflict occurs when a transition (t, c) disables the firing of another transition (t', c'). A sufficient condition is that (t, c) consumes tokens needed by (t', c'). The following mapping computes the set of instances (t', c') disabled by a transition (t, c) via a place p :

$$^tW^-(p, t') \circ (W^-(p, t) \setminus W^+(p, t))(c)$$

As proposed by Dutheillet and Haddad, this notion of conflict can be used to an implementation of simulation. Indeed, given a marking m and a successor m' such that $m[(t, c)\rangle m'$ only the transitions which are in conflict with the fired transition (t, c) are no more firable at m'. For instance let us take net depicted on figure 2. When transition $(t, \langle X_t, Y_t\rangle)$ is fired only those transitions $(t', \langle X_{t'}, Y_{t'}\rangle)$ which satisfy the constraints $X_{t'} = X_t \wedge Y_{t'} = Y_t \wedge \neg(X_{t'} = X_t \wedge Y_{t'} = f(Y_t))$ are removed from the list of enabled transitions. If the net is not safe we still have to check that the transition has been disabled since there may be still enough tokens to fire the transition.

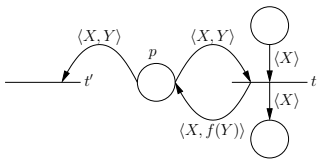

Fig. 2. A conflict between t and t'

5 Conclusion

Because of the management of the color mappings, colored Petri nets introduce an additional complexity in structural analysis techniques. We have defined in this work a framework based on constraints system which enables to deal with this complexity in a symbolic manner. The class of colored nets studied include a large set of color mappings while it still enables analysis possibilities. By two different examples we have proved the relevance of this work and that it can be the basis of an efficient implementation of techniques such as net reductions or simulation.

References

1. Dutheillet C. and Haddad S. An efficient computation of structural relations in unary regular nets. In *Seventh International Symposium on Computer and Information Sciences (ISCIS VII)*, pages 73–79, 1992.

2. Dutheillet C. and Haddad S. Structural analysis of coloured nets. application to the detection of confusion. Technical report, Rapport IBP/MASI, 1992.
3. Dutheillet C. and Haddad S. Conflict sets in colored petri nets. In *5th International Workshop on Petri Nets and Performance Models, Toulouse (F) 19.-22. October 1993*, pages 76–85, 1993.
4. Poitrenaud D. and Pradat-Peyre J.-F. Pre- and post-agglomerations for LTL model checking. In Nielsen, M. and Simpson, D., editors, *Lecture Notes in Computer Science: 21st International Conference on Application and Theory of Petri Nets (ICATPN 2000), Aarhus, Denmark, June 2000*, volume 1825, pages 387–408. Springer-Verlag, 2000.
5. Berthelot G. *Transformations et Analyse de Reseaux de Petri. Applications aux Protocoles.* These d'Etat, Univ. Paris VI, June 1983. NewsletterInfo: 16.
6. Berthelot G. Transformations and decompositions of nets. In Brauer, W., Reisig, W., and Rozenberg, G., editors, *Lecture Notes in Computer Science: Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, September 1986*, volume 254, pages 359–376. Springer-Verlag, 1987. NewsletterInfo: 27.
7. Jensen K. Coloured petri nets: A high level language for system design and analysis. *Lecture Notes in Computer Science; Advances in Petri Nets 1990*, 483:342–416, 1991. NewsletterInfo: 39.
8. Couvreur J. M. and Haddad S. Validation of parallel systems with coloured petri nets. In Cosnard, M. et al., editors, *Parallel Processing. Proceedings of the IFIP WG 10.3 Working Conference, 1988, Pisa, Italy*, pages 377–390, Amsterdam, The Netherlands, 1988. North-Holland.
9. Brgan R. and Poitrenaud D. An efficient algorithm for the computation of stubborn sets of well formed petri nets. In *Proceeding of the 16th International Conference on Application and Theory of Petri Nets, Turin, June 1995.*, pages 121–140, 1995.
10. Evangelista S. Structural analysis of quasi well formed nets. Technical report, Rapport Cedric/CNAM, http://cedric.cnam.fr/, 2004.
11. Haddad S. *Une categorie reguliere de reseau de Petri de haut niveau: definition, proprietes et reductions, application a la validation de systemes distribues.* Thesis, Univ. Paris, France, 1987.
12. Haddad S. A reduction theory for coloured nets. *Lecture Notes in Computer Science; Advances in Petri Nets 1989*, 424:209–235, 1990.
13. Haddad S. and Pradat-Peyre J.-F. Efficient reductions for ltl formulae verification. Technical report, Rapport Cedric/CNAM, http://cedric.cnam.fr/, 2004.